Soldier Station:
Integrating Constructive and Virtual Models

Shirley Pratt and David Pratt
Department of Computer Science, Naval Postgraduate School, Monterey, CA 93943
pratts@cs.nps.navy.mil pratt@cs.nps.navy.mil

David Ohman and John Galloway
TRADOC Analysis Center, ATRC-WAC, White Sands Missile Range, NM 88002
ohman@trac.wsmr.army.mil gallowaj@trac.wsmr.army.mil

1. Abstract

Soldier Station is a unique simulation system which bridges the gap between two distinct realms of modeling: constructive and virtual. The Soldier Station operator controls a simulated dismounted infantry soldier in a 3D virtual environment with rules of movement, engagement and tactics provided from a constructive model. This paper describes the Soldier Station system, its design, and the integration of two separate simulations with radically different modeling philosophies. The features of the resulting system, its limitations, and plans for future work are presented.

2. Introduction

Traditional US Army simulations generally fall into one of two distinct modeling realms: constructive or virtual. Constructive simulations allow a user the ability to control one or more battlefield entities subject to software rules, data and procedures. Entities are indirectly controlled through the user's interactions with a 2D plan view display and a Graphical User Interface (GUI). In contrast, virtual simulations strive to immerse a user into a 3D synthetic environment as the entity itself. The user interacts with various input devices which provide direct control over the entity. Currently, systems of both types of simulations are in popular use. Major efforts have been expended to allow these disparate systems the ability to participate together in joint Distributed Interactive Simulation (DIS) exercises. Soldier Station is a unique effort which brings the two modeling realms together within a single DIS-compatible system.

Soldier Station bridges the gap between the two modeling realms by integrating together the US Army's constructive Janus model algorithms and the Naval Postgraduate School's NPSNET virtual environment system. It allows for visual realism and user interactivity that is currently not available in standard US Army constructive models. It utilizes realistic movement, detection and engagement algorithms not present in most virtual simulators. The integration of two well established simulations represents a significant reduction in project risk while offering significant advantages over building either system independently. The primary purpose of Soldier Station is to serve as an analytic tool for TRADOC Analysis Center (TRAC) to address Land Warrior program issues concerning Dismounted Infantry (DI) command and control, situational awareness, tactics, techniques and procedures.

Soldier Station is DIS-compatible and able to interoperate with other constructive and virtual systems which use DIS Version 2.0.3 or 2.0.4 network protocols. The availability of a particular terrain database format required by an application, however, can be another limiting factor to this interoperatilibity. As part of this project, a terrain tool was developed to convert the gridded Janus terrain data into a polygonal database format appropriate for many visual simulations including NPSNET and Soldier Station. Generally speaking, terrain format incompatibilities are a major problem in the DIS simulation community which exceeds the scope of this paper.

3. System Overview

Soldier Station is actually a system of systems. It is designed to run on two separate Silicon Graphics (SGI) workstations, a multiple processor SGI Onyx Reality Engine2 and a SGI Indy. Two machines are used to accommodate the large graphics and CPU processing requirements of the main simulation system and to meet substantial user interface system demands. One SGI Onyx with a multi-channel option (MCO) is actually more expensive and more likely to become overloaded resulting in poor system performance than a two machine system.

3.1 User Interface Components

Figure 1 shows the various user interface components of Soldier Station. The visual display is a 3D perspective view of the synthetic environment. This view depends on the DI entity's posture, head and body orientation and the current sensor. From two nearby speakers the user can hear DIS networked battlefield sounds. Verbal communication with other DIS participants is possible using a telephone or radio headset.

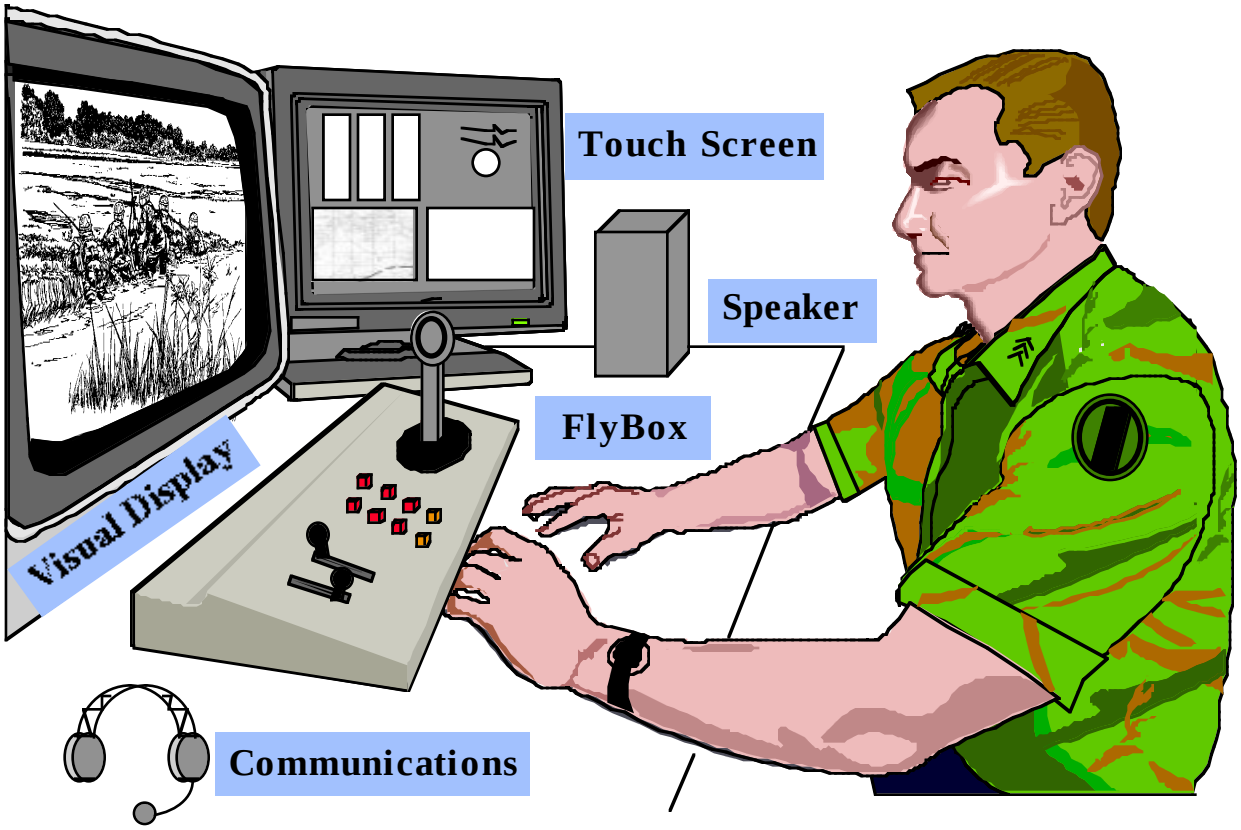

Figure 1: User Interface Components of Soldier Station

The operator controls the input speed and the soldier's posture using levers and switches on the BG Systems Flybox input device. The Flybox joystick has three degrees of freedom allowing body and head/weapon orientation (heading and pitch). It also has a trigger for weapon firing capabilities. From the touch screen GUI the user may easily select different types of weapons, sensors or instruct the DI entity to execute one of numerous different types of hand signals. To orient himself on the battlefield, there is a 2D map which can show various information overlays and the relative position of other detected or dead entities. The GUI also has a compass which indicates the current body and head/weapon orientation, and various textual feedback information (i.e. location coordinates, actual speed of travel, ammunition rounds left, movement/injury status). Table 1 summarizes some of the various options which the Soldier Station operator may select while controlling the DI.

3.2 Software Components

The software components of Soldier Station are shown in Figure 2. The main system runs on the SGI Onyx and is comprised of two tightly coupled modules, a Visualization Module (VM) and a Combat Module (CM). The User Interface System (UIS) runs on the SGI Indy and consists of two separate applications, a GUI and a sound server.

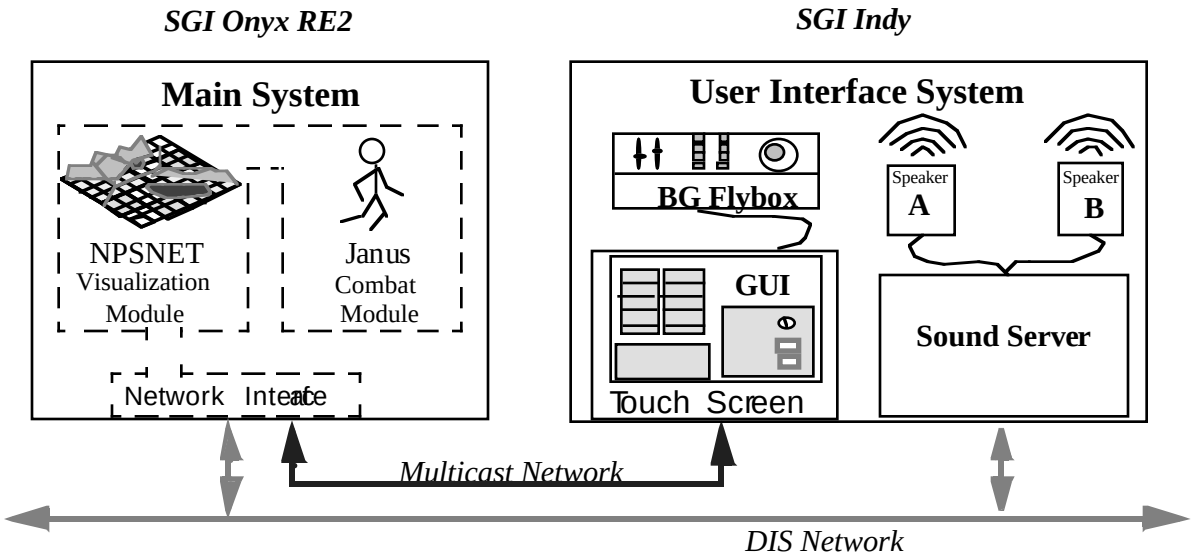

Figure 2: Software Components of Soldier Station

The VM is responsible for overall Soldier Station program control. It is a modified version of Naval Postgraduate School's NPSNET Version IV.8 system (Pratt et al., 1996b). NPSNET is an object oriented C++ application which uses the SGI Performer visual simulation toolkit (Rohlf and Helman, 1994) to create 3D graphical representations of terrain, objects, entities and environmental effects. The VM also provides DIS network management, remote entity dead reckoning (DR) and simulation of the local DI entity. Soldiers are generally represented using a medium resolution, fully-articulated soldier model and a library of real-time animations from University of Pennsylvania's Jack system (Granieri and Badler, 1995).

The Combat Module (CM) is based on the US Army's Janus system (US Army, 1996). It contains the FORTRAN algorithms for Soldier Station system initialization, DI movement, detection, weapon firing and damage assessment. These routines, which were originally part of Verified and Validated (V&V) Janus Version 4.2 code, were modified to allow enhanced user control over the DI entity. Later they were upgraded to Janus Version 6.0 which most notably supports multiple sides of forces and fratricide. Although the modified routines are still subject to the V&V process, they provide realistic feedbacks for maneuvering over terrain and obstacles, for weapons firing outcomes and injury determinations which were not previously present in NPSNET.

The GUI application acts a central collection point for all of the user input made via the BG Flybox and touch screen devices. It packages the inputs into Interface Data Units (IDU) protocols and sends them to the main system via multicasting. The VM processes the inputs, makes the appropriate calls to the CM routines and then sends back feedback data in IDUs for the GUI to display. Thus, it is possible for the operator to request unrealistic speeds, postures or weapon firing given certain situations and be limited by the CM which allows, in theory, only reasonable outcomes to occur.

In addition to the feedback data displayed on the GUI, the user hears various battlefield sounds from an

NPSNET sound server application running on the SGI Indy. The sound server listens for certain DIS PDUs which it recognizes as having a sound associated with them (primarily Fire and Detonation PDUs and also some of the local entity s Entity State PDUs). Upon receiving such PDUs, the distance between the DI entity and the source of the sound is computed as the sound wave propagates in the virtual environment. When the distance is zero, the sound is played over the speakers adding considerable realism to the simulation.

3.3 System Design Considerations

The Soldier Station system is designed to host the VM and CM together as a single integrated application running on the SGI Onyx. The modules interact extensively during the simulation and pass considerable amounts of information between them. In order to minimize the communication latency between the two modules, the CM routines are bundled into a library and linked to the VM.

Substantial user interface requirements for the Soldier Station system justify the use of a second low-end SGI workstation. A separate monitor is needed for a touch screen. The GUI application manages inputs from both the touch screen and the BG Flybox devices, passes user input information to the main system, and receives and displays feedback information from the main system. In addition, the sound server application requires a minimum of 32 MB memory to be able to play sounds instantaneously upon receiving the appropriate PDUs. These demands and the high cost of a new SGI MCO display drove the decision to run the UIS system on a separate SGI Indy workstation. This two machine configuration frees valuable computational resources for the main system which has substantial demands for graphics, entity simulation and networking.

As shown in Figure 2, the main system on the SGI Onyx interacts with the GUI application on the SGI Indy via multicast networking protocols. Although multicasting is inherently an unreliable means of network communication, it provides some attractive features. Namely, it allows multiple Soldier Station suites to co-exist on the same physical network by partitioning the network traffic into separate multicast groups which each suite can subscribe to (Pratt et al., 1996a). The ability to subscribe to multiple multicast groups, if desired, also allows the future development of a logger application which can record multicast network traffic for user analysis purposes.

4. System Development

The integration of two distinct simulation systems, Janus and NPSNET, was much easier said (and drawn) than done. The integrated system was envisioned to work together seamlessly, however, the two component systems are based on radically different modeling philosophies, are written in different computer languages, and utilize different terrain file formats. The integration was primarily carried out in an stepwise manner with often multiple iterations occurring at each step:

```
do
  merge code
  test results
  do
    debug problems
    test results
  until CORRECT
until DONE
```

4.1 Integrating Janus and NPSNET

Integration of the two modules which comprise the main Soldier Station system required careful coordination and consideration between the VM and CM developers. The first step was to identify exactly what the types of interactions were sought between the VM and CM. This step produced five main CM driver routines as listed in Table 2 (Ohman, 1996).

4.1.1 Modifications to Janus

The appropriate constructive model algorithms for DIs were isolated from the original Janus code. The CM continues to use the exact same input data files as Janus (i.e. FORCE, DEPLOY, JSCRN, and SYSTEM) and has complete knowledge of all entity attributes and system characteristics. These CM routines control one interactive DI entity so at least one system type in the FORCE file must be the same as the Soldier Station system type specified during program startup. The CM must inform the VM of the DI entity system type s capabilities in order for the user to be able to effectively control the entity. Thus, SS_setup passes back the names of the weapons and sensors along with the starting number of ammunition rounds available for each program run.

Some modifications to the Janus routines were introduced to resolve time and space incompatibilities and to provide enhanced user control. Since Janus is an event driven simulation, changes were made to allow the CM algorithms to be called in real-time and re-

gardless of the frame rate. Built-in time delays for the soldier's movements due to obstacles and suppression by fire were shortened from minutes to seconds in order to allow the user to respond in a realistic amount of time to such situations. All references to nodes for routes and movement control were removed with input now being provided by the Soldier Station operator via the Flybox. The DI entity may move in one direction and look in another by having his head turned. He may also now move backwards so that the soldier can remain facing forward while retracing his last steps instead of having to turn around. He can also now enter any infantry foxhole or vehicle prepared fighting position regardless of which Janus side it belongs to.

Unlike traditional Janus DIs, the Soldier Station entity can fire his weapon on command at no specific targets (e.g. generate suppressive fire), and also fire in a non-horizontal plane. He can detect up to twenty-five targets (instead of just ten as for Janus DIs) provided he is alive and does not have a major wound which might impact his ability to detect targets. If targets are detected, the CM allows the Soldier Station to fire at them regardless of side assuming that other firing criteria have been met. Thus, the Soldier Station entity can always possibly commit fratricide if he does not exercise good judgment in the synthetic battlefield. As in Janus, the firing criteria which must simultaneously be met include meeting minimum safe range requirements for firing a weapon, and meeting weapon jam/clearing times and ammunition reloading times. Short firing delays are introduced if any of these criteria are not met.

Along with added control the user is also faced with more potential simulation hazards. For example, the entity is now also able to step on and detonate detected mines whereas Janus entities can not detonate mines previously detected. Automatic defilade status changes have also been removed so that the DI entity no longer changes from fully exposed to partial defilade when he stops moving unless directed by the user to do so. In fact, the DI entity may remain fully exposed while being fired upon if the user does not take any action.

For demonstration purposes, the DI entity is also allowed to be resurrected if killed. In addition to possibly being killed or suppressed, the soldier may also now be wounded. For determining a wound status, the body is divided into six parts, each having a probability of being wounded depending upon the current posture. The damage assessment routine processes direct fire, indirect fire or mine explosion events.

4.1.2 Modifications to NPSNET

To integrate the CM with the VM, NPSNET program flow was examined to determine where and how the CM driver routines should be called. Since Soldier Station was designed to use the NPSNET visual simulation framework, the relatively minor modifications were needed to be able to interact with them. These included making coordinate conversions to/from NPS coordinates to Janus UTM coordinates and transferring information from C++ dynamic data structures into static arrays to pass to the CM. Code was also added to map internal NPSNET vehicle numbers of remote entities to Janus unit numbers. These Janus unit numbers are pre-defined in the Janus FORCE file which is read when SS_setup is called during system startup. Remote DIS entities must pass an appropriate Janus unit number in DIS Entity State PDU markings fields to the VM in order to be recognized by the CM as valid units (a requirement which will hopefully be removed in the near future).

Strict interfaces for each of the CM driver routines were defined, e.g. function name and the number, type and order of the arguments. C++ wrappers (which account for the C++ name mangling of function names and facilitate correct argument type passing) were then created so that the CM functions could be called directly from the VM. The five main CM driver routines and numerous other supporting routines are archived together in a library object and linked to the VM.

There were several features added to NPSNET to comply with the additional capabilities required for Soldier Station. To allow the Soldier Station entity the ability to select between up to five different weapons, additional weapon models (besides the existing M16 rifle) were added with only one showing at any given time. Additional sensor views for the binocular and gun sights sensors were added as 2D overlays over the 3D view along with concurrent changes in the field of view. To receive the user's inputs from the UIS, multicast IDUs were defined.

Support for up to six different sides of DIs was provided. This involved creating DI models with numerous different colored uniforms and one model which carried no weapon (to be used as a civilian). Low resolution soldier models were also incorporated for low level detections which assume that features such as uniform patterns, faces, objects carried and even limbs are not visible. Presently, only DIs have multiple levels of detail models available.

4.2 Conversion of Janus Terrain

The terrain data formats used by Janus and visual systems such as NPSNET are completely different. The terrain elevations in Janus are gridded and assumed to be constant within each grid square (or pixel based). If represented in 3D graphics, the terrain would appear as a collection of cubes with different heights. On the other hand, in NPSNET the terrain file format is polygonal in nature and its representation consists of triangles connecting each grid point to the next grid point forming a relatively smooth terrain surface when compared to Janus' terrain representation.

The relatively small size of the DI entities demands smooth 3D terrain representations in order to avoid large visual abnormalities in the terrain database at the edge of each Janus grid square. However, the CM routines still represent the terrain as gridded cells internally. Thus, there is a mismatch in the internal terrain representation for each module which sometimes causes the DI entity to appear either above or below the actual polygonal ground surface. Some detection mismatches would also likely occur causing entities to disappear/appear although the VM may not actually show terrain blocking/not blocking the line of sight. Both of these problems are more predominant in terrain areas where steep and/or rapidly varying gradients exist.

As part of the NPSNET-Janus integration, a SGI-based software tool was developed to convert the Janus terrain into a MultiGen Flight format for the VM. The terrain tool reads in the Janus gridded terrain elevations and the separate polygonal Janus feature data (vegetation, roads, rivers, buildings, etc.) and converts them into the required polygonal MultiGen format. Since the CM uses the Janus terrain format while the VM uses the polygonal database format, the Soldier Station system relies on this ability to convert various Janus terrain databases. With the availability of MultiGen terrain databases, other DIS-compatible simulations which also use the format are also be able to interoperate with Soldier Station. So far, Soldier Station has successfully participated in numerous exercises with remote DIS entities generated from Janus linked to DIS (JLINK) (Pate and Roussos, 1996), NPSNET, and another Soldier Station system.

The terrain tool is capable of automatically handling most types of Janus terrain databases with minimal user intervention. Because of the very large number of grid points present in most Janus terrain databases, the tool subsamples the points, using every other grid point, in an effort to reduce the number of polygons. The time it takes to convert a terrain database depends strongly on the number of Janus polygonal terrain features as well as on the number of elevation grid points present. A medium sized terrain database (say, 10 km by 10 km) with an average number of polygonal features takes about ten minutes to convert. Currently, small tree areas with relatively large concave edges can cause some conversion problems.

4.3 Main Program Flow

Figure 3 is a simplified program flow chart of the integrated NPSNET-Janus main system. Essentially, various initializations are carried out and then several major tasks are carried out continuously as long as the simulation continues. These tasks include the handling of DIS network PDUs, handling user input IDUs, simulation of the local DI entity, updating the position and statuses of the remote entities and finally drawing the scene. It is noted, however, that usually network management and drawing would be handled asynchronously in a multiprocessing mode.

During program startup, the VM calls SS_setup once. Various Janus data files are read and, if desired, post processor data files are started to log data from the Soldier Station during the simulation. In the simulation loop, PDUs from other remote entities on the DIS network are processed. User input IDUs from the GUI are processed similarly, and given the current inputs, SS_move is called to determine the soldier s next position and his movement status. Remote entities are updated using dead reckoning, and once a second SS_search is called to determine what live entities the Soldier Station entity can detect. Dead entities are not detected by Janus and are not passed to SS_search. However, they are still displayed on the GUI 2D map as black icons and appear visibly damaged on the synthetic battlefield.

When an IDU packet contains data about a trigger pull effected, SS_reload is called to determine the outcome of the firing event. If the weapon was successfully fired, a fire PDU is sent out over the DIS network. If a close proximity detonation PDU is received from another remote site, SS_assess is called to determine the extent of an injury, if any. If the soldier is uninjured, the simulation proceeds as before. If the soldier is killed, he can be resurrected by the user in demo mode, or the user can elect to exit the simulation. If the soldier is killed or wounded, movement, detection and/or firing capabilities will be impaired depending on the injury. For simplicity, Figure 3 assumes that the DI entity is uninjured.

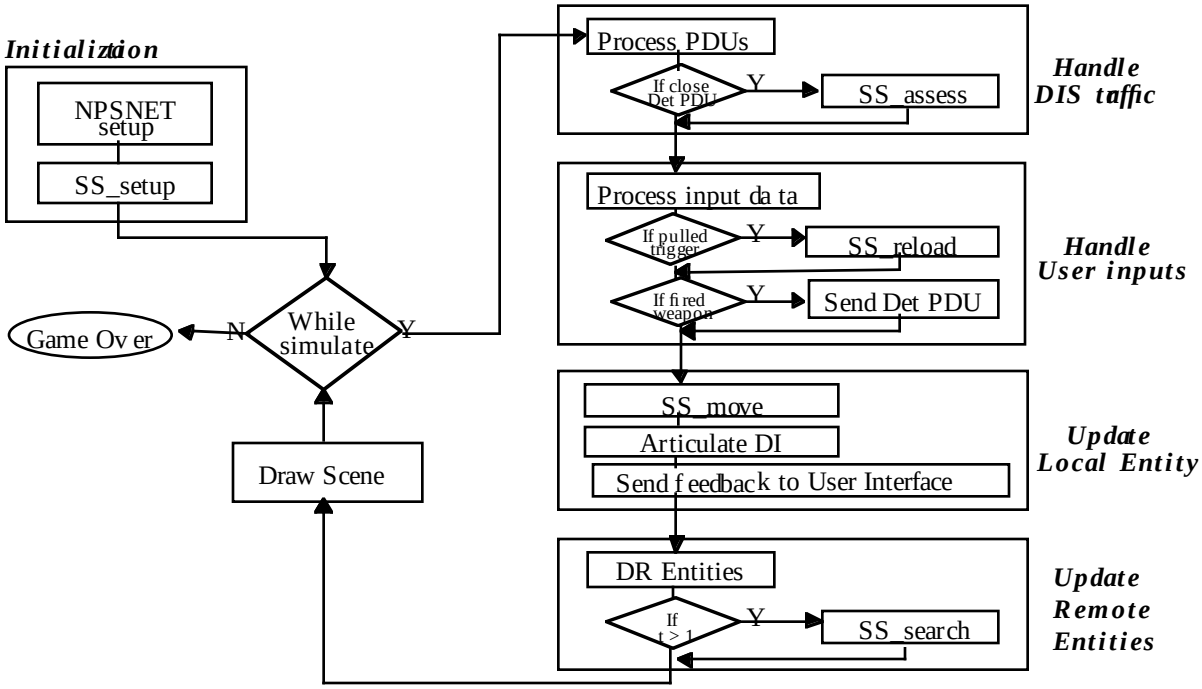

Figure 3: Simplified Main Program Flow

During each frame, feedback data about the current status of the DI entity and what entities have been detected at what detection level (aimpoint, recognition and identification) is sent back to the GUI. This feedback information is then displayed on the GUI and could affect the user's next inputs. Table 3 lists some of the possible outcomes from making function calls to the CM driver routines.

The program flow shown in Figure 3 is actually nearly the same as the normal program flow of NPSNET with the exception of the calls to CM routines. Without the CM driver routines, NPSNET assumes that the DI entity can always move regardless of terrain characteristics, can detect anything that is drawn, can fire upon anything when the trigger is pulled, and is always fatally wounded by any close proximity detonation. Clearly, the integration of the Janus algorithms brings much needed realism into the visual simulation.

5. Conclusions

The decision to merge Janus and NPSNET together to form one seamless Soldier Station system was, to a large extent, governed by practical reasons. Namely, the analysis needs of the Soldier Station project could be met while significantly reducing project development time and costs by integrating two existing systems rather than developing either one independently. By reusing code from NPSNET, Soldier Station acquired a major head start on underlying 3D graphics, basic entity simulation, DIS networking and sound requirements. New development efforts could be focused on the integration with Janus, adding necessary features which were not currently available in NPSNET and the development of the UIS. By merging Janus algorithms, the virtual simulation acquired robust mobility characteristics and target detections as well as realistic weapon firing outcomes and injury assessments. These features replaced non-existent or very simplistic (and generally unrealistic) graphics based capabilities which were previously available in NPSNET.

The combined system, represents a significant improvement over either of its component parts, but it does have some limitations. Currently the representation of terrain, soldier movements and engagements, and visual parameters are at moderate levels of detail. These are subject to change according to the resolution needed by the simulation, but large, high resolution terrain databases with many entities (say, more than fifty) present will degrade the system performance without further optimizations (as mentioned below). The inherently different VM and CM inter-

nal terrain representations causes some visual inconsistencies which need to be resolved. For improved DIS-compatibility, unit numbers should be able to be created or assigned dynamically within the CM. The transfer of data from VM's dynamic data structures to static arrays for the CM routines is unavoidable without major code changes to the VM data structures. However, this could impact performance otherwise and requires some in-depth system performance analyses beforehand. Some data transfers are simply unavoidable due to C++ and FORTRAN language differences.

For increased system performance, we plan to spawn a separate process to obtain the computationally expensive CM detection routine output asynchronously via shared memory buffers. Optimizations to the visual simulation include an upgrade to SGI Performer 2.x which supports terrain database paging and increased use of level of detail modeling techniques. To significantly lower hardware system costs, we plan to tune the main system to run on the new, much less expensive SGI Maximum Impact workstations. Enhancements for Soldier Station to participate in night time and urban environment simulations are also planned in the near future.

6. Acknowledgments

The authors would like to thank Mr. David Ward, CPT Steve Brown, CPT Bill Smith, MAJ Glen Roussos, Mr. David Hastings, LTC Ralph Wood, COL Carl Baxley (Ret) and Mr. Roy Reynolds for their help and valuable inputs to the system. Development and demonstrations of the system would not have been possible without the support of TRAC-WSMR, TRAC-MTRY and NPS computer systems personnel and students.

7. References

Granieri, J. and Badler, N. (1995). Simulating Humans in VR. To appear in R. Earnshaw, J. Vince, and H. Jones, editors. *Applications of Virtual Reality*, Academic Press.

Naval Postgraduate School, Computer Science Department (1996). NPSNET Research Group Internet Home Page *http://www-npsnet.cs.nps.navy.mil/npsnet*.

Ohman, D. (1996) *Soldier Station Combat Module Documentation*. Prepared for TRAC-WSMR by Nations, Inc.

Pate, M., and Roussos, G. (1996) JLINK - A Distributed Interactive Janus. *Phalanx*, Vol. 29, No. 1, 12-15.

Pratt, D., Barham, P., Barker, R., and McMillan, S. (1996a) AUSA 95 DI Demonstration. *14th DIS Workshop Proceedings,* 1165 - 1170.

Pratt, S., Pratt, D., Waldrop, M., Barham, P., Ehlert, J., and Chrislip, C. (1996b) Humans in Large-scale, Real-time, Networked Virtual Environments. Submitted for publication in *Presence.*

Rohlf, J., and Helmann, J. (1994). IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics. *SIGGRAPH '94 Proceedings in Computer Graphics*, 381-394.

U.S. Army TRADOC Analysis Center, White Sands Missile Range, NM (1996) Janus Version 6.0 Documentation.

8. Authors Biographies

Shirley Pratt is a Computer Scientist in the Computer Science Department at Naval Postgraduate School (NPS). She is the lead developer of the NPSNET visualization module for the Soldier Station system. Ms. Pratt has a M.S. in Ocean Physics from the U.C. San Diego and a B.A. in Applied Mathematics from U.C. Berkeley. Her research interests include real-time modeling of the environment, virtual simulations of dismounted infantry, and the use of efficient DIS aggregation protocols.

CW4 David Ohman (Ret) is an Operations Research/Systems Analyst working for Nations, Inc. in support of the Soldier Station project for TRADOC Analysis Center, White Sands Missile Range (TRAC-WSMR). Prior to joining Nations, he was the only military member assigned as a Software Developer for the Janus Interactive Simulation Development Division at TRAC-WSMR. He has an M.S. in Industrial Engineering from the University of Texas, El Paso, an M.A. in Management from Webster University, and a B.S. in Business and Management from the University of Maryland. He is also a graduate of the US Army Operations Research/Systems Analysis Military Applications course.

David Pratt is serving as the first Technical Director of the Joint Simulation System (JSIMS) Joint Project Office in Orlando, Florida. He holds this position concurrently with an appointment as a tenure track faculty member at the Department of Computer Science, NPS. Prior to joining the faculty at NPS, Dr. Pratt was a Data Processing Officer in the United States Marine Corps. He holds a Ph.D. and a M.S. in Computer Science from NPS and a B.S. in Electrical Engineering from Duke University. His research interests include distributed simulation and

architectures to support scalability in real-time 3D computer graphics.

John Galloway is an Operations Research Analyst for TRAC-WSMR. Since joining TRAC-WSMR in 1986, he has been involved with Combined Arms Support Task Force Evaluation Model (CASTFOREM), Janus and Soldier Station. He is the Program Leader for the Soldier Station project and has worked within the dismounted infantry arena for constructive models for the past four years. He has a B.S. in Civil Engineering from New Mexico State University.

Control Item	Available Options	User Interface
Posture	Upright, Crouching, Kneeling, Prone, Fox hole, Deploy weapon, Align head and body orientation, Lase target	Flybox Buttons
Sensor	Eye balls, Binoculars, Gun Sights (Thermal in future)	GUI Radio Buttons
Weapon	M16A2 Rifle, M203 grenade launcher, M60 machine gun, M249 semi-automatic weapon, M72 light anti-tank weapon (Object Individual Combat Weapon in future)	GUI Radio Buttons
Hand Signals	Various signals for movement control, formations, fire control, emergency alerts, echelon designation, and other miscellaneous signals	GUI Push Buttons
Map Display	Map displayed, not displayed Zoom in, zoom out Show topographical contour shading, grid lines, terrain features, buildings, obstacles Detected entity icons shown, not shown	GUI Push Buttons

Table 1: User Selectable Options for Soldier Station

CM Routine	Purpose	Example Considerations
SS_setup	Reads Janus data. Passes DI capabilities back to VM	Scenario, run, system number inputs
SS_move	Determines current location, actual speed, movement status	Soldier posture / orientation, terrain characteristics, requested speed, suppression status, wound status
SS_detect	Determines what live entities are visible at what detection level	Soldier posture / wound status, active sensor, target defilade status/speed, LOS probability
SS_reload	Determines firing result, impact point, rounds remaining	Soldier posture / orientation, target type, active weapon, rounds left, time last fired
SS_assess	Determines injury, if any, due to a close proximity detonation	Soldier posture, munition type, impact location, firing entity, luck

Table 2: Description of the Five CM Driver Routines

CM Routine	Basic Outputs	Specific Information
SS_move	Movement status	Moving on a road, in vegetation, in an urban area, in a river
	Speed status	Moving at requested speed, slowed by terrain, moving at maximum speed allowed, not moving because soldier is kneeling or in a fox hole
	Delay status	Obstructed by a building, fence, another unit, by a river, an abatii, a smoke pot, a mine
	Posture status	In the requested posture, not in a fox hole because none nearby
SS_detect	No entities detected	None
	Entities detected	For each entity, detection at level: aim point = 1, recognition = 2, identification = 3
SS_reload	Successfully fired	Fired at a target, or generated suppressive fire Impact point is XYZ
	Unsuccessfully fired	Unable to fire because: Soldier is wounded, moving too fast, being suppressed, not ready to fire Range is too far, too close Weapon is out of ammunition, pointed at too large of a pitch angle Target is a non-combatant, is dead, a low probability hit, a bad target, an identified friendly target
SS_assess	Injury status	Not injured, dead, wounded
	Wound type	Hit in the head, chest, stomach, pelvis, leg, arm
	Suppression status	Not suppressed, suppressed by fire

Table 3: Example Feedback Data from theCM